

## Chapter 2: Calculators

### Chapter Summary

This chapter explains basic structures of programming which any language can have. Each structure is followed by example code.

In this chapter, we will write a code for Simple Calculator, Trigonometric Calculator, Log Calculator and then Grade Calculator. We will cover almost all basic structure of programming.

### Constant

Constant is the name of memory location with fixed value.

```
Public Const E As Double = 200000000000.0  
Private Const E As Double = 200000000000.0  
Protected Friend Const E As String = 200000000000.0
```

### Variables

It is the name of memory location where we put our data. We use variable for data manipulation / mathematical calculation. Declaring variable, use **Dim** keyword. Sometimes declared with default value assigned.

```
Dim a as integer  
Dim b as double = 0
```

In first line of above code, integer variable with the name of a is declared. In the second line, double variable with the name of b and initialized with the default value of 0

Basic variable types are:

1. Numeric
  - a. Integer
  - b. Single
  - c. Double
  - d. Decimal
  - e. Long
2. String
3. Date/Time
4. Boolean
5. Object

Integers are used for whole numbers. Single is used for floating point number. Double is used for floating point number with greater accuracy w.r.t. Single. Long is used for financial calculation. String and char is used for Text variable. Object is used for reference of classes / controls. Date / time is clear from its name



<b>floating-point)</b>			4.94065645841246544E-324 through 1.79769313486231570E+308 † for positive values
<b>Integer</b>	Int32	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
<b>Long (long integer)</b>	Int64	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807 (9.2...E+18 †) (signed)
<b>Object</b>	Object (class)	4 bytes on 32-bit platform	Any type can be stored in a variable of type Object
		8 bytes on 64-bit platform	
<b>SByte</b>	SByte	1 byte	-128 through 127 (signed)
<b>Short (short integer)</b>	Int16	2 bytes	-32,768 through 32,767 (signed)
<b>Single (single-precision floating-point)</b>	Single	4 bytes	-3.4028235E+38 through -1.401298E-45 * for negative values;
			1.401298E-45 through 3.4028235E+38 * for positive values
<b>String (variable-length)</b>	String (class)	Depends on implementing platform	0 to approximately 2 billion Unicode characters
<b>UInteger</b>	UInt32	4 bytes	0 through 4,294,967,295 (unsigned)
<b>ULong</b>	UInt64	8 bytes	0 through 18,446,744,073,709,551,615 (1.8...E+19 *) (unsigned)
<b>User-Defined (structure)</b>	(inherits from ValueType)	Depends on implementing platform	Each member of the structure has a range determined by its data type and independent of the ranges of the other members
<b>UShort</b>	UInt16	2 bytes	0 through 65,535 (unsigned)

\* In *scientific notation*, "E" refers to a power of 10. So 3.56E+2 signifies  $3.56 \times 10^2$  or 356, and 3.56E-2 signifies  $3.56 / 10^2$  or 0.0356.

## ***Scope of Variable***

Variable's scope is defined as the declaration space in which no other variable can have the same name. In scope, variable can be called with out its full name or use of import statement. In some cases, the variable's access level can influence its scope.

<b>Level</b>	<b>Description</b>
<b>Block scope</b>	Available only within the code block in which it is declared
<b>Procedure scope</b>	Available to all code within the procedure in which it is declared
<b>Module scope</b>	Available to all code within the module, class, or structure in which it is declared
<b>Namespace scope</b>	Available to all code in the namespace in which it is declared

### **Block Scope**

A block is a set of statements enclosed within initiating and terminating declaration statements, such as the following:

1. Do and Loop
2. For [Each] and Next
3. If and End If
4. Select and End Select
5. SyncLock and End SyncLock
6. Try and End Try
7. While and End While
8. With and End With

## ***Life Time***

The lifetime of a declared element is the period of time during which it is available for use.

Variables are the only elements that have lifetime. For this purpose, the compiler treats procedure parameters and function returns as special cases of variables.

The lifetime of a variable represents the period of time during which it can hold a value. Its value can change over its lifetime, but it always holds some value.

---

***Note: Lifetime has: Beginning, End and Extension***

---

## ***Access Level***

The access level of a declared element is the extent of the ability to access it, that is, what code has permission to read it or write to it.

This is determined not only by how you declare the element itself, but also by the access level of the element's container.

---

**Note:** Access level is control by declaration keywords like *Dim*, *Private*, *Protected*, *Friend*, *Protected Friend* or *Public*.

---

### Public

The Public (Visual Basic) keyword in the declaration statement specifies that the elements can be accessed from code anywhere in the same project, from other projects that reference the project, and from any assembly built from the project. The following code shows a sample Public declaration.

```
Public Class classForEverybody
```

You can use Public only at module, interface, or namespace level. This means you can declare a public element at the level of a source file or namespace, or inside an interface, module, class, or structure, but not in a procedure.

### Protected

The Protected (Visual Basic) keyword in the declaration statement specifies that the elements can be accessed only from within the same class, or from a class derived from this class. The following code shows a sample Protected declaration.

```
Protected Class classForMyHeirs
```

You can use Protected only at class level, and only when you declare a member of a class. This means you can declare a protected element in a class, but not at the level of a source file or namespace, or inside an interface, module, structure, or procedure.

### Friend

The Friend (Visual Basic) keyword in the declaration statement specifies that the elements can be accessed from within the same assembly, but not from outside the assembly. The following code shows a sample Friend declaration.

```
Friend stringForThisProject As String
```

You can use Friend only at module, interface, or namespace level. This means you can declare a friend element at the level of a source file or namespace, or inside an interface, module, class, or structure, but not in a procedure.

### Protected Friend

The Protected and Friend keywords together in the declaration statement specify that the elements can be accessed either from derived classes or from within the same assembly, or both. The following code shows a sample Protected Friend declaration.

```
Protected Friend stringForProjectAndHeirs As String
```

You can use Protected Friend only at class level, and only when you declare a member of a class. This means you can declare a protected friend element in a class, but not at the level of a source file or namespace, or inside an interface, module, structure, or procedure.

### Private

The Private (Visual Basic) keyword in the declaration statement specifies that the elements can be accessed only from within the same module, class, or structure. The following code shows a sample Private declaration.

```
Private numberForMeOnly As Integer
```

You can use Private only at module level. This means you can declare a private element inside a module, class, or structure, but not at the level of a source file or namespace, inside an interface, or in a procedure.

At the module level, the Dim statement without any access level keywords is equivalent to a Private declaration. However, you might want to use the Private keyword to make your code easier to read and interpret.

### Simple Calculator

1. Go to **File > New Project** and select **Windows Forms Application**. Change the name to **SimpleCalc**. Press **OK**. A form will appear.
2. Our simple calculator will take the input of two numbers and perform basic arithmetic operations like add, minus, multiply, divide, power and remainder.

---

**Note:** We use label for display or for result, textbox for input and button for execution of commands.

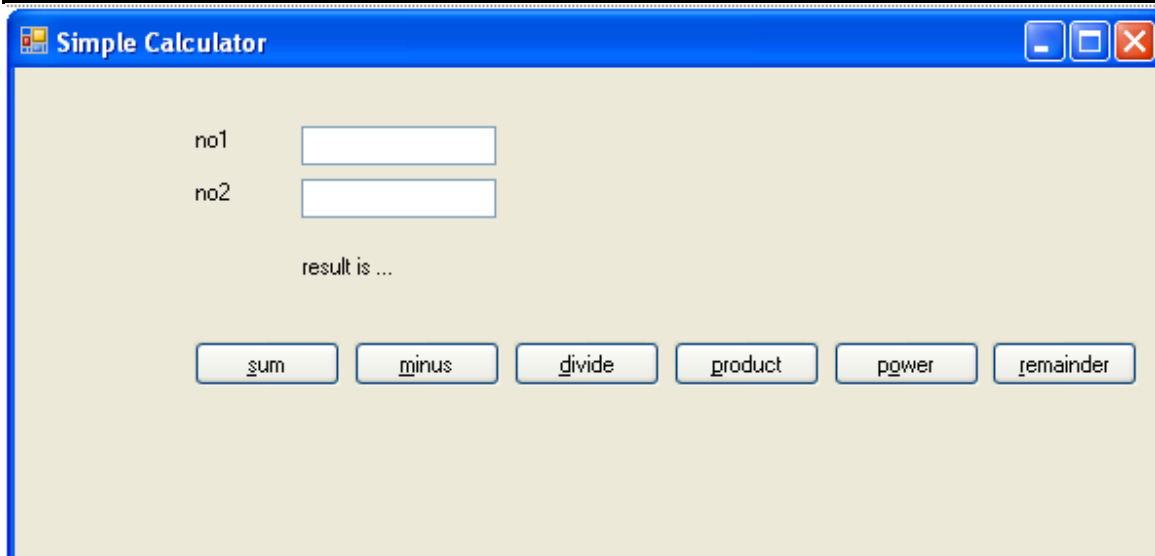
---

3. Add 2 textboxes, 3 labels and 6 buttons, then, change their properties as per table below:

Sno	Control	Property	Value
1	Label1	Text	no1
2	Label2	Text	no2
3	Label3	Text	result is ...
4	Button1	Text	&sum
5	Button2	Text	&minus
6	Button3	Text	&divide
7	Button4	Text	&product

8	Button5	Text	p&ower
9	Button6	Text	&remainder

**Note:** Use of Ampersand sign (&) in text property enables the control to be accessed by Alt key at run time



4. Double click the button1 and add the following code of sum to the click event so that we have:

```
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
Dim r As Double = no1 + no2
Label3.Text = r.ToString
```

5. Press **F5** to run the program. Enter 2 numbers in the given textboxes and then press **Alt+s** or click the sum button to calculate the result.
6. Double click the button5 and add the following code of power to the click event so that we have:

```
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
Dim r As Double = no1 ^ no2
r = Math.Round(r, 5)
Label3.Text = r.ToString
```

You can also use **math.pow(no1,no2)** instead of **no1^no2** to calculate the power.  
**Math.round(r,5)** rounds the number r to 5 decimal places.

7. Similarly double click the button6 and add the following code of remainder to the click event so that we have:

```
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
Dim r As Double = no1 Mod no2
Label3.Text = r.ToString
```

8. Rest of the code for minus, divide, and product is same as that of sum but only with change of operator. Final code is given below:

```
Public Class Form1
```

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
```

```
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
Dim r As Double = no1 + no2
Label3.Text = r.ToString
```

```
End Sub
```

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
```

```
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
Dim r As Double = no1 - no2
Label3.Text = r.ToString
```

```
End Sub
```

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
```

```
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
Dim r As Double = no1 / no2
Label3.Text = r.ToString
```

```
End Sub
```

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
```

```
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
```

```
Dim r As Double = no1 * no2
Label3.Text = r.ToString

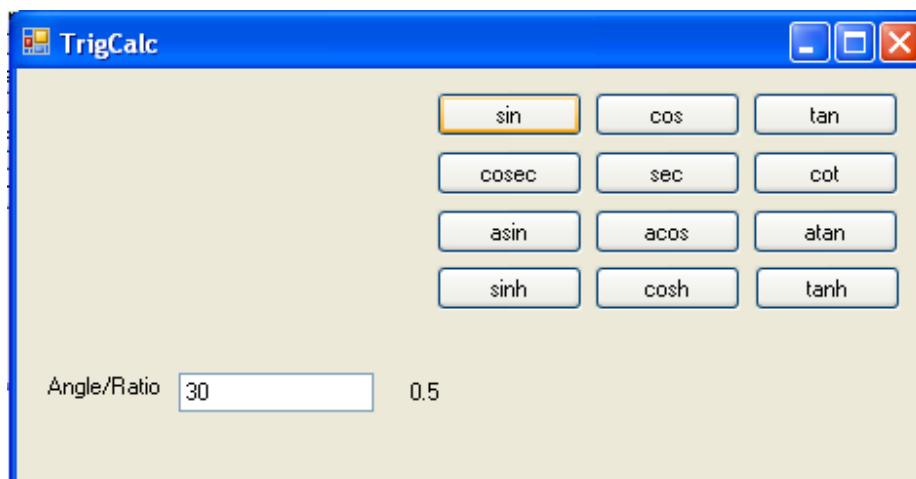
End Sub
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button5.Click
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
Dim r As Double = no1 ^ no2
r = Math.Round(r, 5)
Label3.Text = r.ToString

End Sub

Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button6.Click
Dim no1 As Double = Val(TextBox1.Text)
Dim no2 As Double = Val(TextBox2.Text)
Dim r As Double = no1 Mod no2
Label3.Text = r.ToString
End Sub

End Class
```

## Trigonometric Calculator



Sno	Control	Property	Value
1	Label1	Text	Angle/Ratio
2	Label2	Text	Result is ...
4	Button1	Text	sin
5	Button2	Text	cos
6	Button3	Text	tan
7	Button4	Text	cosec
8	Button5	Text	sec
9	Button6	Text	cot
10	Button7	Text	asin
11	Button8	Text	acos
12	Button9	Text	atan
13	Button10	Text	sinh
14	Button11	Text	cosh
15	Button12	Text	tanh

*Public Class Form1*

*Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click*

*Dim ang As Double = Val(TextBox1.Text)*

*Dim sinr As Double = 0*

*ang = Math.PI / 180 \* ang*

*sinr = Math.Sin(ang)*

*sinr = Math.Round(sinr, 5)*

*Label2.Text = sinr.ToString*

*End Sub*

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
```

```
    Dim ang As Double = Val(TextBox1.Text)
    Dim cosr As Double = 0
    ang = Math.PI / 180 * ang
    cosr = Math.Cos(ang)
    cosr = Math.Round(cosr, 5)
    Label2.Text = cosr.ToString
```

```
End Sub
```

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
```

```
    Dim ang As Double = Val(TextBox1.Text)
    Dim tanr As Double = 0
    ang = Math.PI / 180 * ang
    tanr = Math.Tan(ang)
    tanr = Math.Round(tanr, 5)
    Label2.Text = tanr.ToString
```

```
End Sub
```

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
```

```
    Dim ang As Double = Val(TextBox1.Text)
    Dim cosecr As Double = 0
    ang = Math.PI / 180 * ang
    cosecr = 1 / Math.Sin(ang)
    cosecr = Math.Round(cosecr, 5)
    Label2.Text = cosecr.ToString
```

```
End Sub
```

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
```

```
    Dim ang As Double = Val(TextBox1.Text)
    Dim secr As Double = 0
    ang = Math.PI / 180 * ang
    secr = 1 / Math.Cos(ang)
    secr = Math.Round(secr, 5)
    Label2.Text = secr.ToString
```

```
End Sub
```

```
Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button6.Click
```

```
    Dim ang As Double = Val(TextBox1.Text)
    Dim cotr As Double = 0
```

```
    ang = Math.PI / 180 * ang
    cotr = 1 / Math.Tan(ang)
    cotr = Math.Round(cotr, 5)
    Label2.Text = cotr.ToString
End Sub

Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button7.Click
    Dim rto As Double = Val(TextBox1.Text)
    Dim asina As Double = 0
    asina = Math.Asin(rto)
    asina = 180 / Math.PI * asina
    asina = Math.Round(asina, 5)
    Label2.Text = asina.ToString

End Sub

Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button8.Click
    Dim rto As Double = Val(TextBox1.Text)
    Dim acosa As Double = 0
    acosa = Math.Acos(rto)
    acosa = 180 / Math.PI * acosa
    acosa = Math.Round(acosa, 5)
    Label2.Text = acosa.ToString

End Sub

Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button9.Click
    Dim rto As Double = Val(TextBox1.Text)
    Dim atana As Double = 0
    atana = Math.Atan(rto)
    atana = 180 / Math.PI * atana
    atana = Math.Round(atana, 5)
    Label2.Text = atana.ToString

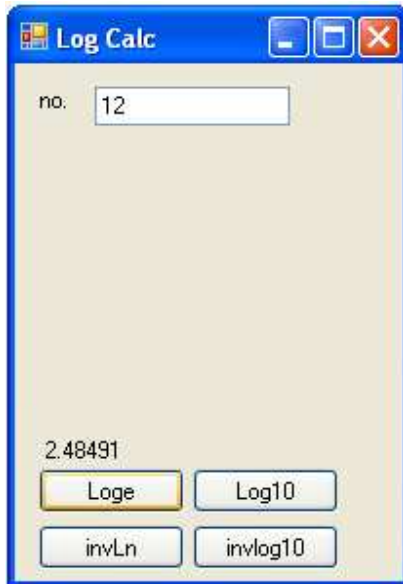
End Sub

Private Sub Button10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button10.Click
    Dim angh As Double = Val(TextBox1.Text)
    Dim sinhr As Double = 0
    angh = Math.PI / 180 * angh
    sinhr = Math.Sinh(angh)
    sinhr = Math.Round(sinhr, 5)
    Label2.Text = sinhr.ToString
End Sub
```

```
Private Sub Button11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button11.Click
    Dim angh As Double = Val(TextBox1.Text)
    Dim coshr As Double = 0
    angh = Math.PI / 180 * angh
    coshr = Math.Cosh(angh)
    coshr = Math.Round(coshr, 5)
    Label2.Text = coshr.ToString
End Sub

Private Sub Button12_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button12.Click
    Dim angh As Double = Val(TextBox1.Text)
    Dim tanhr As Double = 0
    angh = Math.PI / 180 * angh
    tanhr = Math.Tanh(angh)
    tanhr = Math.Round(tanhr, 5)
    Label2.Text = tanhr.ToString
End Sub
End Class
```

## Log Calculator



Sno	Control	Property	Value
1	Label1	Text	no
2	Label2	Text	Result is ...
3	Button1	Text	Loge
4	Button2	Text	Log10
5	Button3	Text	invLn
6	Button4	Text	invlog10

```
Public Class Form1
```

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
Button1.Click
```

```
        Dim no As Double = Val(TextBox1.Text)
```

```
        Dim ln As Double = Math.Log(no)
```

```
        ln = Math.Round(ln, 5)
```

```
        Label2.Text = ln.ToString
```

```
    End Sub
```

```
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
Button2.Click
```

```
        Dim no As Double = Val(TextBox1.Text)
```

```
        Dim log10 As Double = Math.Log10(no)
```

```
        log10 = Math.Round(log10, 5)
```

```
        Label2.Text = log10.ToString
```

```
    End Sub
```

```
    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
Button3.Click
```

```
        Dim ln As Double = Val(TextBox1.Text)
```

```
        Dim invln As Double = Math.Exp(ln)
```

```
        invln = Math.Round(invln, 5)
```

```
        Label2.Text = invln.ToString
```

```
    End Sub
```

```
    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
Button4.Click
```

```
        Dim log10 As Double = Val(TextBox1.Text)
```

```
        'Dim invlog10 As Double = Math.Pow(10, log10)
```

```
        Dim invlog10 As Double = 10 ^ log10
```

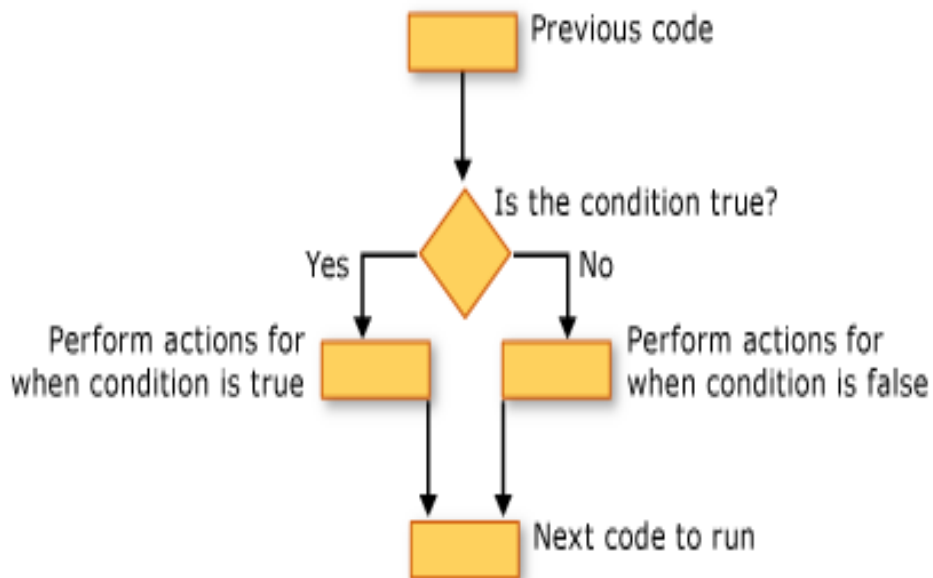
```
        invlog10 = Math.Round(invlog10, 5)
```

```
        Label2.Text = invlog10.ToString
```

```
    End Sub
```

```
End Class
```

## ***Branching and Decision Structure***



1. If
  - a. If then End if
  - b. If then Else
  - c. If then Elseif
2. Select Case
3. Try Catch
4. Switch
5. On Error
6. Goto
7. Choose
8. Switch

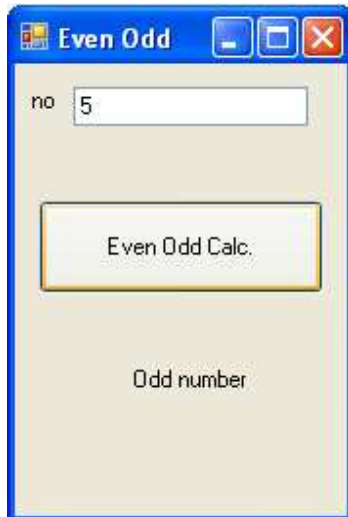
If structure is written as:

```
If condition [ Then ]  
  [ statements ]  
[ Elseif elseifcondition [ Then ]  
  [ elseifstatements ] ]  
[ Else
```

```

    [ elsestatements ]
End If
-or-
If condition Then [ statements ] [ Else [ elsestatements ] ]

```



Sno	Control	Property	Value
1	Label1	Text	no
2	Label2	Text	Result is ...
3	Button1	Text	Even Odd Calc

```
Public Class Form1
```

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
```

```
        Dim no As Double = Val(TextBox1.Text)
```

```
        If no Mod 2 = 0 Then
```

```
            Label2.Text = "Even number"
```

```
        Else
```

```
            Label2.Text = "Odd number"
```

```
        End If
```

```
    End Sub
```

```
End Class
```

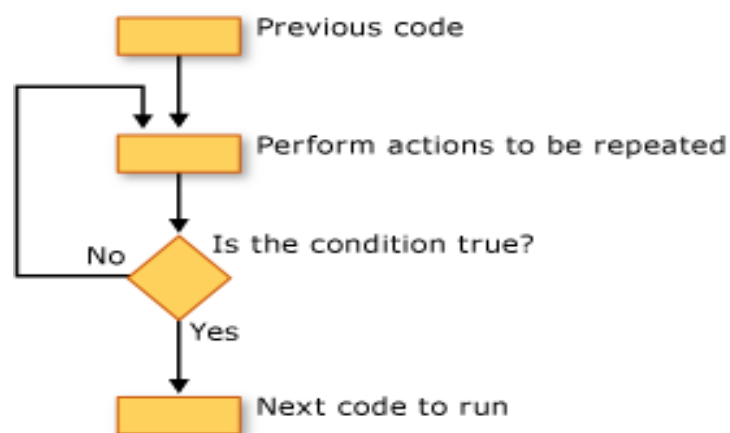
The structure of select case is written as:

```
Select [ Case ] testexpression  
[ Case expressionlist  
[ statements ] ]  
[ Case Else  
[ elstatements ] ]  
End Select
```

Example code is:

```
Dim S As String = txtCode.text  
Select case S  
    Case "A", "a"  
        Msgbox "America"  
    Case "B", "b"  
        Msgbox "Brazil"  
    Case else  
        Msgbox "Unknown Code"  
End select
```

## Loops



1. While Loop
2. Do Loop
3. For Loop
4. For Each Loop
5. Recursive Function

### While Loop

Runs a series of statements as long as a given condition is True.

```
While condition  
[ statements ]  
[ Exit While ]  
[ statements ]  
End While
```

```
Dim counter As Integer = 0  
While counter < 20  
    counter += 1  
    debug.writeline counter
```

```
'if counter = 10 then exit while  
End While  
MsgBox("While loop ran " & CStr(counter) & " times")
```

### Do Loop

Repeats a block of statements while a Boolean condition is True or until the condition becomes True.

```
Do { While | Until } condition  
[ statements ]  
[ Exit Do ]  
[ statements ]  
Loop
```

-or-

```
Do  
    [ statements ]  
[ Exit Do ]  
[ statements ]  
Loop { While | Until } condition
```

In below structure, the loop will run once irrespective of the condition.

```
Debug.WriteLine("Do Demo 1:")  
Do while i <= 10  
    debug.writeline i  
    i = i+1  
Loop  
  
Debug.WriteLine("Do Demo 2:")  
Do  
    debug.writeline i  
    i = i+1  
Loop while i <= 10
```

### For Next Loop

Repeats a group of statements a specified number of times.

```
For counter [ As datatype ] = start To end [ Step step ]  
[ statements ]  
[ Exit For ]  
[ statements ]  
Next [ counter ]
```

```
For i = 1 to 10  
    Debug.writeline i  
Next i  
  
For i = 1 to 10 step 2  
    Debug.writeline i  
Next i  
  
For i = 1 to 10 step -1  
    Debug.writeline i  
'if i = 8 then exit for  
Next i
```

### Code for Prime Composite number:

```
Dim n As Double = 0  
n = CDb(txtNo.Text)  
For i As Integer = n - 1 To 2 Step -1  
    If n Mod i = 0 Then  
        MsgBox("Composite number!")  
        Exit Sub  
    End If  
Next  
MsgBox("Prime number!")
```

### For Each Loop:

Repeat a group of statements for each element in collection.

```
For Each element [ As datatype ] In group  
    [ statements ]  
[ Exit For ]  
[ statements ]
```

Next [ element ]

Following are two examples to use for each loop in VB.NET. You can call these procedures from a button click.

```
Private Sub ForEachEx()  
  
    Dim ds As DataSet = New DataSet("dsFood")  
    Dim tbl As DataTable = New DataTable("tblFruits")  
    ds.Tables.Add(tbl)  
  
    ' Create new DataColumn, set DataType, ColumnName  
    ' and add to DataTable.  
    Dim col = New DataColumn()  
    col.DataType = System.Type.GetType("System.String")  
    col.ColumnName = "colFruits"  
    col.ReadOnly = False  
    col.Unique = False  
  
    ' Add the Column to the DataColumnCollection.  
    tbl.Columns.Add(col)  
  
    Dim row0 As DataRow = tbl.NewRow()  
    row0(col) = "Apple"  
    tbl.Rows.Add(row0)  
  
    Dim row1 As DataRow = tbl.NewRow()  
    row1(col) = "Mango"  
    tbl.Rows.Add(row1)  
  
    Dim row2 As DataRow = tbl.NewRow()  
    row2(col) = "Banana"  
    tbl.Rows.Add(row2)  
  
    Dim row3 As DataRow = tbl.NewRow()  
    row3(col) = "Orange"  
    tbl.Rows.Add(row3)  
  
    Dim row4 As DataRow = tbl.NewRow()  
    row4(col) = "Strawberry"  
    tbl.Rows.Add(row4)  
  
    ' Loop through each row in the view.  
  
    For Each r As DataRow In tbl.Rows
```

```
    If r(col) = "Strawberry" Then
        MsgBox("Found it")

    End If
Next

End Sub

Private Sub ForEachEx1()
    Dim found As Boolean = False
    Dim thisCollection As New Collection
    thisCollection.Add("Apple")
    thisCollection.Add("Mango")
    thisCollection.Add("Banana")
    thisCollection.Add("Orange")
    thisCollection.Add("Strawberry")

    For Each thisObject As String In thisCollection
        If thisObject = "Strawberry" Then
            Debug.Write(thisObject)
            found = True
            MsgBox("Found the Strawberry")
            Exit For
        End If
    Next thisObject

End Sub
```

## Recursive Function

Self calling function is called recursive function

```
Function factorial(ByVal n As Integer) As Integer
    If n <= 1 Then
        Return 1
    Else
        Return factorial(n - 1) * n
    End If
End Function
```

## Grade Calculator

Sno	Control	Property	Value
1	Label1	Text	Obtained Marks
2	Label2	Text	Maximum Marks
3	Label3	Text	Letter Grade is:
4	Label4	Text	Grade Point is:
5	Button1	Text	&Grade

#### Public Class Form1

*Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click*

*Dim om As Double = Val(TextBox1.Text)*  
*Dim mm As Double = Val(TextBox2.Text)*

*If om < 0 Or om > 100 Then*

*MsgBox("Enter obtained marks within range of 0 - 100")*  
*Exit Sub*

*Elseif mm < 0 Or mm > 100 Then*

*MsgBox("Enter Maximum marks within range of 0 - 100")*  
*Exit Sub*

*Elseif om > mm Then*

*MsgBox("Enter Obtained marks less or equal to Maximum marks")*  
*Exit Sub*

*End If*

*Dim rm As Double = om / mm \* 100*  
*rm = Math.Ceiling(rm)*

```
If rm >= 96 Then
    Label3.Text = "Letter Grade is: A"
    Label4.Text = "Grade Point is: 4.00"
Elseif rm >= 91 Then
    Label3.Text = "Letter Grade is: A-"
    Label4.Text = "Grade Point is: 3.67"
Elseif rm >= 86 Then
    Label3.Text = "Letter Grade is: B+"
    Label4.Text = "Grade Point is: 3.33"
Elseif rm >= 81 Then
    Label3.Text = "Letter Grade is: B"
    Label4.Text = "Grade Point is: 3.00"
Elseif rm >= 76 Then
    Label3.Text = "Letter Grade is: B-"
    Label4.Text = "Grade Point is: 2.67"
Elseif rm >= 71 Then
    Label3.Text = "Letter Grade is: C+"
    Label4.Text = "Grade Point is: 2.33"
Elseif rm >= 66 Then
    Label3.Text = "Letter Grade is: C"
    Label4.Text = "Grade Point is: 2.00"
Elseif rm >= 61 Then
    Label3.Text = "Letter Grade is: C-"
    Label4.Text = "Grade Point is: 1.67"
Elseif rm >= 56 Then
    Label3.Text = "Letter Grade is: D+"
    Label4.Text = "Grade Point is: 1.33"
Elseif rm >= 51 Then
    Label3.Text = "Letter Grade is: D"
    Label4.Text = "Grade Point is: 1.00"
Else
    Label3.Text = "Letter Grade is: F"
    Label4.Text = "Grade Point is: 0.00"
End If
End Class
```

Instead of If then Elseif, we can use select case as follows:

```
Select Case rm
    Case Is >= 96
        Label3.Text = "Letter Grade is: A"
        Label4.Text = "Grade Point is: 4.00"
    Case Is >= 91
        Label3.Text = "Letter Grade is: A-"
        Label4.Text = "Grade Point is: 3.67"
    Case Is >= 86
```

```
Label3.Text = "Letter Grade is: B+"
Label4.Text = "Grade Point is: 3.33"
Case Is >= 81
Label3.Text = "Letter Grade is: B"
Label4.Text = "Grade Point is: 3.00"
Case Is >= 76
Label3.Text = "Letter Grade is: B-"
Label4.Text = "Grade Point is: 2.67"
Case Is >= 71
Label3.Text = "Letter Grade is: C+"
Label4.Text = "Grade Point is: 2.33"
Case Is >= 66
Label3.Text = "Letter Grade is: C"
Label4.Text = "Grade Point is: 2.00"
Case Is >= 61
Label3.Text = "Letter Grade is: C-"
Label4.Text = "Grade Point is: 1.67"
Case Is >= 56
Label3.Text = "Letter Grade is: D+"
Label4.Text = "Grade Point is: 1.33"
Case Is >= 51
Label3.Text = "Letter Grade is: D"
Label4.Text = "Grade Point is: 1.00"
Case Else
Label3.Text = "Letter Grade is: F"
Label4.Text = "Grade Point is: 0.00"
End Select
```

### Sub procedure

Sub-Procedure is a block of code which do operation and do not report any result or

Any code within sub / end sub is called sub procedure

Sub are within block of sub / end sub

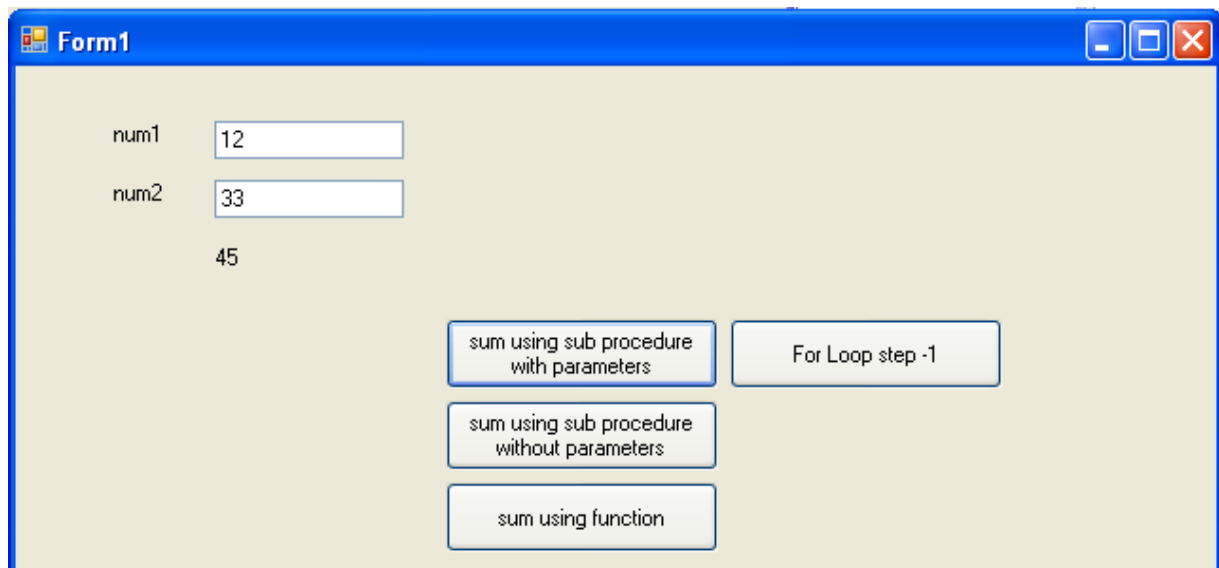
Statements within sub / end sub should be limited to a dozen line of code

Sub may or may not accept arguments. Arguments can be passed by value or by reference. In by value, original variables are used whose value may get changed. ByVal arguments are faster and available in default. In by reference, copy of variable is used in caluclations. ByRef arguments do not changes in value but they are slower compared to byVal.

### Functions

Function is a procedure which do required operation and reports the result as well. Or code within Function / End Function is called a functions. Function can be equated to some variable. These are also called mini-programs.

### Example of Sub Procedure and Function



The screenshot shows a Windows form titled "Form1" with a light beige background. On the left side, there are two text boxes: "num1" containing the value "12" and "num2" containing the value "33". Below these is a label displaying the number "45". On the right side, there are four buttons arranged in a vertical column. The top button is labeled "sum using sub procedure with parameters" and has a blue border. The second button is labeled "sum using sub procedure without parameters". The third button is labeled "sum using function". The fourth button, positioned to the right of the first three, is labeled "For Loop step -1".

Sno	Control	Property	Value
1	Label1	Text	num1
2	Label2	Text	num2
3	Label3	Text	Result is ...
4	Button1	Text	sum using sub procedure with parameters
5	Button2	Text	Sum using sub procedure without parameters
5	Button3	Text	sum using function
5	Button4	Text	For Loop step -1

*Public Class Form1*

*'to be used for sub with paramenters and function call*

*Dim no1, no2 As Double*

*Dim functionR As Double*

*Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click*

*'to be used for sub without paratmeters*

*Call sum1()*

*End Sub*

*Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click*

*'to be used for sub with parameters*

*Call getInput()*

*Call sum2(no1, no2)*

*End Sub*

*Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click*

*'to be used with function call*

*getInput()*

*functionR = sum3(no1, no2)*

*Label3.Text = functionR.ToString*

*End Sub*

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button4.Click
    Dim sR As String = String.Empty

    For i As Integer = 10 To 1 Step -1
        sR &= i.ToString & vbCrLf
    Next i
    Label3.Text = sR
End Sub

'Demo of sub procedure without parameters
Sub sum1()
    Dim no1, no2, r As Double
    no1 = Val(TextBox1.Text)
    no2 = Val(TextBox2.Text)
    r = no1 + no2
    Label3.Text = r.ToString

End Sub

'Demo of sum with parameters
Sub sum2(ByVal x As Double, ByVal y As Double)
    Dim r As Double = x + y
    Label3.Text = r.ToString

End Sub
'Input to variables
Sub getInput()

    no1 = Val(TextBox1.Text)
    no2 = Val(TextBox2.Text)
End Sub

'Demo of Function
Function sum3(ByVal x As Double, ByVal y As Double) As Double
    Return x + y
End Function

End Class
```

***Reference:***

1. Microsoft Developer Network – MSDN <http://msdn.microsoft.com/>
2. VB.NET Help